R Multivariate Regression Sandbox

R Markdown and Notebooks

This is an R Studio Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

(Caution: Typos are Legion)

When in our school's Math Statistics courses, R is a typical platform that will be used in the classroom.

R is open source and can be aquired here.

https://www.r-project.org

Since R is open source and is commonly used by a broad community it has gotten a large following. Also since it's open source there are a HUGE number of discipline-specific packages. (Many of them don't necessarly do statistics but since R is free…)

A common workbench that is used with R is R-Studio which also has an open-source version that you can download from here.

This demonstrates using R to preform a multivariate regression on the Concrete Dataset

https://www.rstudio.com

R Studio also has support to bring in the many libraries that are available.

This particular example leverages this dataset.

http://kyrill.ias.sdsmt.edu/cee_284/L20_Non_Linear_Regression_Sandbox.xlsx

You can download it to your machine and make a small edit further down to access it with R.

You will need the following packages and their dependencies whcih can be installed with R Studio before you start.

- XLConnect [The XLConnect Package] for reading spreadsheets,
- MASS [Functions and datasets to support Venables and Ripley, "Modern Applied Statistics with S"]. and
- plyr [Tools for Splitting, Applying and Combining Data]
- e1071 [Misc Functions of the Department of Statistics, Probability Theory Group] for getting skewness and kurtosis

to access most of the linear regression activities here.

And to do some of the fancy plots you;ll see you will want this one:

- beanplot [Visualization via Beanplots (like Boxplot/Stripchart/Violin Plot)]

Warning XLConnect will run off of Java and is therefore often a memory hog and a little pokey for large files but it gives you more control than other Excel R tools

This segment of the R code allows you to access these packages.

(Notice the # character. In R, the # denotes the start of a comment and can be on the same line as your code)

```
library(XLConnect) # Load the "Excel Connector for R" Library. (to load the spreadsheet data)
```

```
## Loading required package: XLConnectJars
```

```
## XLConnect 0.2-14 by Mirai Solutions GmbH [aut],
##   Martin Studer [cre],
##   The Apache Software Foundation [ctb, cph] (Apache POI),
##   Graph Builder [ctb, cph] (Curvesapi Java library)

## http://www.mirai-solutions.com
## https://github.com/miraisolutions/xlconnect
```

```
library(MASS)      # Load the "Modern Applied Statistics with S Library"" (for regressions)
library(plyr)      # Load the "Tools for Splitting, Applying and Combining Data" Library
library(e1071)     # Load the "Misc Functions of the Department of Statistics, Probability Theory Group
library(beanplot)  # Load the "Visualization via Beanplots" Library
```

Fetching, Reading and Prepping our Dataset

Now let's get the file holding our concrete data.

First download the excel work book found here.

http://kyrill.ias.sdsmt.edu/cee_284/L20_Non_Linear_Regression_Sandbox.xlsx

You wil

```
excel_file_name <- "/Users/wjc/Downloads/L20_Non_Linear_Regression_Sandbox.xlsx"
```

Import the data from the spreadsheet into a single data "frame" which works like a table or ledger of data.

This command uses the "readWorksheetFromFile()" function which is loaded with the "XLConnect" package.

(Also when I use functions in R, I tend to get very anal and try to comment the arguments as best as I can since I don't use R on a daily basis.)

In this case we are using a spreadsheet with multiple pages.

Our concrete data is on the sheet called "Concrete."

We are pulling data in from a specific region of the sheet (including a header line) which goes from A2 to K105.

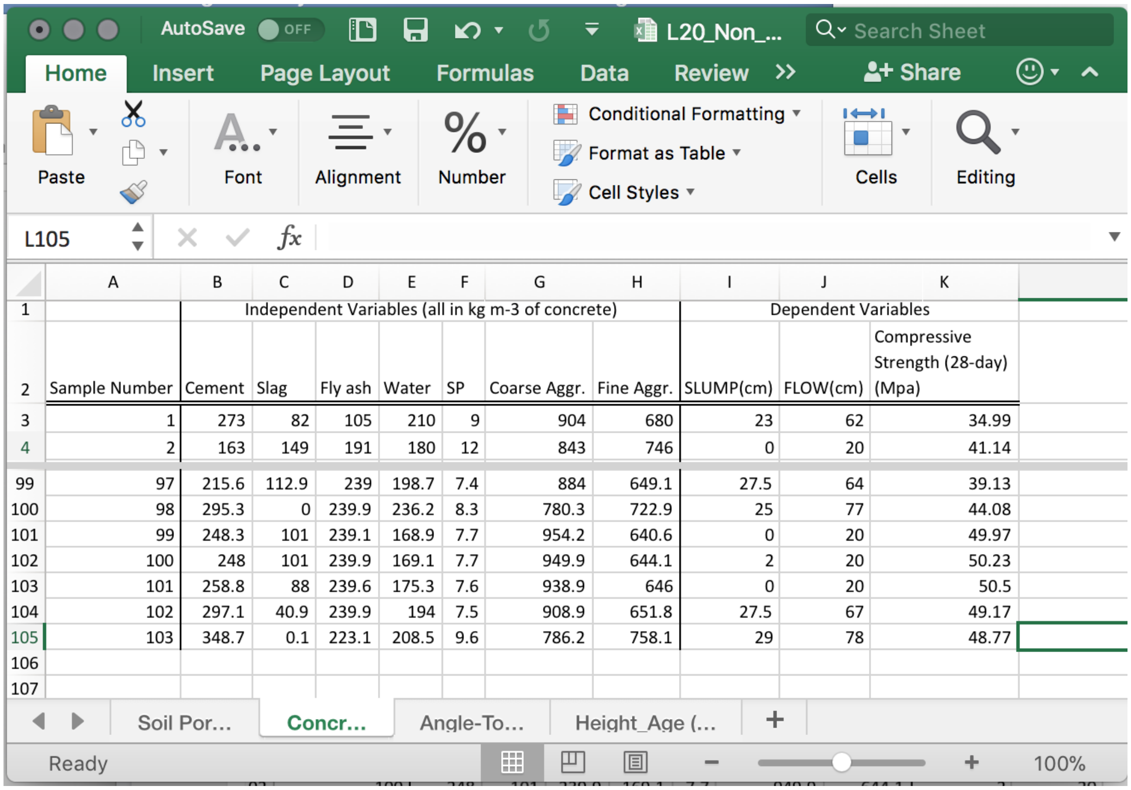
Here we will grab data from the sheet and call the whole group of data (the data "frame") "exceldata" with the function readWorksheetFromFile()

```
exceldata <- readWorksheetFromFile(file   = excel_file_name, # file name here
                                   sheet  = "Concrete",      # select the Concrete Sheet
                                   region = "A2:K105",       # chose the cells you need to import
                                   header = TRUE             # TRUE if the first imported row is the He
                                   )
```

Now with the data loaded let's take a look at the inventory of the data we just imported.

```
str(object = exceldata)
```

```
## 'data.frame':    103 obs. of  11 variables:
##  $ Sample.Number                     : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Cement                            : num  273 163 162 162 154 147 152 145 152 304 ...
##  $ Slag                              : num  82 149 148 148 112 89 139 0 0 0 ...
##  $ Fly.ash                           : num  105 191 191 190 144 115 178 227 237 140 ...
##  $ Water                             : num  210 180 179 179 220 202 168 240 204 214 ...
##  $ SP                                : num  9 12 16 19 10 9 18 6 6 6 ...
##  $ Coarse.Aggr.                      : num  904 843 840 838 923 860 944 750 785 895 ...
##  $ Fine.Aggr.                        : num  680 746 743 741 658 829 695 853 892 722 ...
```



	A	B	C	D	E	F	G	H	I	J	K
1		Independent Variables (all in kg m-3 of concrete)							Dependent Variables		
2	Sample Number	Cement	Slag	Fly ash	Water	SP	Coarse Aggr.	Fine Aggr.	SLUMP(cm)	FLOW(cm)	Compressive Strength (28-day) (Mpa)
3	1	273	82	105	210	9	904	680	23	62	34.99
4	2	163	149	191	180	12	843	746	0	20	41.14
99	97	215.6	112.9	239	198.7	7.4	884	649.1	27.5	64	39.13
100	98	295.3	0	239.9	236.2	8.3	780.3	722.9	25	77	44.08
101	99	248.3	101	239.1	168.9	7.7	954.2	640.6	0	20	49.97
102	100	248	101	239.9	169.1	7.7	949.9	644.1	2	20	50.23
103	101	258.8	88	239.6	175.3	7.6	938.9	646	0	20	50.5
104	102	297.1	40.9	239.9	194	7.5	908.9	651.8	27.5	67	49.17
105	103	348.7	0.1	223.1	208.5	9.6	786.2	758.1	29	78	48.77
106											
107											

Figure 1: Screenshot of the Excel Sheet.

```
## $ SLUMP.cm.                             : num  23 0 1 3 20 23 0 14.5 15.5 19 ...
## $ FLOW.cm.                              : num  62 20 20 21.5 64 55 20 58.5 51 51 ...
## $ Compressive.Strength..28.day...Mpa.: num  35 41.1 41.8 42.1 26.8 ...
```

In looking at this output, we see that the labels (or headers) are not the same as our spreadsheet.

Fix that we can fix that… with the rename() function (which is in the plyr package)

(And we can make a few more tweaks along the way)

```
exceldata <- rename(x       = exceldata,           # data frame you want to patch
                    replace = c("SP" =             # the name you want to change
                          "Superplasticizer")  # the replacement name
                    )

exceldata <- rename(x       = exceldata,
                    replace = c("Fine.Aggr." =
                          "Fine.Aggregates")
                    )

exceldata <- rename(x       = exceldata,
                    replace = c("Coarse.Aggr." =
                          "Coarse.Aggregates")
                    )

exceldata <- rename(x       = exceldata,
                    replace = c("SLUMP.cm." =
                          "SLUMP")
```

```
                   )

exceldata <- rename(x       = exceldata,
                   replace = c("FLOW.cm." =
                                 "FLOW")
                   )

exceldata <- rename(x       = exceldata,
                   replace = c("Compressive.Strength..28.day...Mpa." =
                                 "Compressive.Strength")
                   )
```

OK now let's look at what we just fixed…

```
exceldata
```

```
##      Sample.Number Cement  Slag Fly.ash Water Superplasticizer
## 1                1  273.0  82.0   105.0 210.0              9.0
## 2                2  163.0 149.0   191.0 180.0             12.0
## 3                3  162.0 148.0   191.0 179.0             16.0
## 4                4  162.0 148.0   190.0 179.0             19.0
## 5                5  154.0 112.0   144.0 220.0             10.0
## 6                6  147.0  89.0   115.0 202.0              9.0
## 7                7  152.0 139.0   178.0 168.0             18.0
## 8                8  145.0   0.0   227.0 240.0              6.0
## 9                9  152.0   0.0   237.0 204.0              6.0
## 10              10  304.0   0.0   140.0 214.0              6.0
## 11              11  145.0 106.0   136.0 208.0             10.0
## 12              12  148.0 109.0   139.0 193.0              7.0
## 13              13  142.0 130.0   167.0 215.0              6.0
## 14              14  354.0   0.0     0.0 234.0              6.0
## 15              15  374.0   0.0     0.0 190.0              7.0
## 16              16  159.0 116.0   149.0 175.0             15.0
## 17              17  153.0   0.0   239.0 200.0              6.0
## 18              18  295.0 106.0   136.0 206.0             11.0
## 19              19  310.0   0.0   143.0 168.0             10.0
## 20              20  296.0  97.0     0.0 219.0              9.0
## 21              21  305.0 100.0     0.0 196.0             10.0
## 22              22  310.0   0.0   143.0 218.0             10.0
## 23              23  148.0 180.0     0.0 183.0             11.0
## 24              24  146.0 178.0     0.0 192.0             11.0
## 25              25  142.0 130.0   167.0 174.0             11.0
## 26              26  140.0 128.0   164.0 183.0             12.0
## 27              27  308.0 111.0   142.0 217.0             10.0
## 28              28  295.0 106.0   136.0 208.0              6.0
## 29              29  298.0 107.0   137.0 201.0              6.0
## 30              30  314.0   0.0   161.0 207.0              6.0
## 31              31  321.0   0.0   164.0 190.0              5.0
## 32              32  349.0   0.0   178.0 230.0              6.0
## 33              33  366.0   0.0   187.0 191.0              7.0
## 34              34  274.0  89.0   115.0 202.0              9.0
## 35              35  137.0 167.0   214.0 226.0              6.0
## 36              36  275.0  99.0   127.0 184.0             13.0
## 37              37  252.0  76.0    97.0 194.0              8.0
## 38              38  165.0 150.0     0.0 182.0             12.0
```

```
## 39             39  158.0   0.0   246.0 174.0              7.0
## 40             40  156.0   0.0   243.0 180.0             11.0
## 41             41  145.0 177.0   227.0 209.0             11.0
## 42             42  154.0 141.0   181.0 234.0             11.0
## 43             43  160.0 146.0   188.0 203.0             11.0
## 44             44  291.0 105.0     0.0 205.0              6.0
## 45             45  298.0 107.0     0.0 186.0              6.0
## 46             46  318.0 126.0     0.0 210.0              6.0
## 47             47  280.0  92.0   118.0 207.0              9.0
## 48             48  287.0  94.0   121.0 188.0              9.0
## 49             49  332.0   0.0   170.0 160.0              6.0
## 50             50  326.0   0.0   167.0 174.0              6.0
## 51             51  320.0   0.0   163.0 188.0              9.0
## 52             52  342.0 136.0     0.0 225.0             11.0
## 53             53  356.0 142.0     0.0 193.0             11.0
## 54             54  309.0   0.0   142.0 218.0             10.0
## 55             55  322.0   0.0   149.0 186.0              8.0
## 56             56  159.0 193.0     0.0 208.0             12.0
## 57             57  307.0 110.0     0.0 189.0             10.0
## 58             58  313.0 124.0     0.0 205.0             11.0
## 59             59  143.0 131.0   168.0 217.0              6.0
## 60             60  140.0 128.0   164.0 237.0              6.0
## 61             61  278.0   0.0   117.0 205.0              9.0
## 62             62  288.0   0.0   121.0 177.0              7.0
## 63             63  299.0 107.0     0.0 210.0             10.0
## 64             64  291.0 104.0     0.0 231.0              9.0
## 65             65  265.0  86.0   111.0 195.0              6.0
## 66             66  159.0   0.0   248.0 175.0             12.0
## 67             67  160.0   0.0   250.0 168.0             12.0
## 68             68  166.0   0.0   260.0 183.0             13.0
## 69             69  320.0 127.0   164.0 211.0              6.0
## 70             70  336.0 134.0     0.0 222.0              6.0
## 71             71  276.0  90.0   116.0 180.0              9.0
## 72             72  313.0 112.0     0.0 220.0             10.0
## 73             73  322.0 116.0     0.0 196.0             10.0
## 74             74  294.0 106.0   136.0 207.0              6.0
## 75             75  146.0 106.0   137.0 209.0              6.0
## 76             76  149.0 109.0   139.0 193.0              6.0
## 77             77  159.0   0.0   187.0 176.0             11.0
## 78             78  261.0  78.0   100.0 201.0              9.0
## 79             79  140.0   1.4   198.1 174.9              4.4
## 80             80  141.1   0.6   209.5 188.8              4.6
## 81             81  140.1   4.2   215.9 193.9              4.7
## 82             82  140.1  11.8   226.1 207.8              4.9
## 83             83  160.2   0.3   240.0 233.5              9.2
## 84             84  140.2  30.5   239.0 169.4              5.3
## 85             85  140.2  44.8   234.9 171.3              5.5
## 86             86  140.5  61.1   238.9 182.5              5.7
## 87             87  143.3  91.8   239.8 200.8              6.2
## 88             88  194.3   0.3   240.0 234.2              8.9
## 89             89  150.4 110.9   239.7 168.1              6.5
## 90             90  150.3 111.4   238.8 167.3              6.5
## 91             91  155.4 122.1   240.0 179.9              6.7
## 92             92  165.3 143.2   238.3 200.4              7.1
```

```
## 93             93  303.8   0.2   239.8 236.4            8.3
## 94             94  172.0 162.1   238.5 166.0            7.4
## 95             95  172.8 158.3   239.5 166.4            7.4
## 96             96  184.3 153.4   239.2 179.0            7.5
## 97             97  215.6 112.9   239.0 198.7            7.4
## 98             98  295.3   0.0   239.9 236.2            8.3
## 99             99  248.3 101.0   239.1 168.9            7.7
## 100           100  248.0 101.0   239.9 169.1            7.7
## 101           101  258.8  88.0   239.6 175.3            7.6
## 102           102  297.1  40.9   239.9 194.0            7.5
## 103           103  348.7   0.1   223.1 208.5            9.6
##     Coarse.Aggregates Fine.Aggregates SLUMP FLOW Compressive.Strength
## 1               904.0           680.0 23.00 62.0                34.99
## 2               843.0           746.0  0.00 20.0                41.14
## 3               840.0           743.0  1.00 20.0                41.81
## 4               838.0           741.0  3.00 21.5                42.08
## 5               923.0           658.0 20.00 64.0                26.82
## 6               860.0           829.0 23.00 55.0                25.21
## 7               944.0           695.0  0.00 20.0                38.86
## 8               750.0           853.0 14.50 58.5                36.59
## 9               785.0           892.0 15.50 51.0                32.71
## 10              895.0           722.0 19.00 51.0                38.46
## 11              751.0           883.0 24.50 61.0                26.02
## 12              768.0           902.0 23.75 58.0                28.03
## 13              735.0           836.0 25.50 67.0                31.37
## 14              959.0           691.0 17.00 54.0                33.91
## 15             1013.0           730.0 14.50 42.5                32.44
## 16              953.0           720.0 23.50 54.5                34.05
## 17             1002.0           684.0 12.00 35.0                28.29
## 18              750.0           766.0 25.00 68.5                41.01
## 19              914.0           804.0 20.50 48.2                49.30
## 20              932.0           685.0 15.00 48.5                29.23
## 21              959.0           705.0 20.00 49.0                29.77
## 22              787.0           804.0 13.00 46.0                36.19
## 23              972.0           757.0  0.00 20.0                18.52
## 24              961.0           749.0 18.00 46.0                17.19
## 25              883.0           785.0  0.00 20.0                36.72
## 26              871.0           775.0 23.75 53.0                33.38
## 27              783.0           686.0 25.00 70.0                42.08
## 28              871.0           650.0 26.50 70.0                39.40
## 29              878.0           655.0 16.00 26.0                41.27
## 30              851.0           757.0 21.50 64.0                41.14
## 31              870.0           774.0 24.00 60.0                45.82
## 32              785.0           721.0 20.00 68.5                43.95
## 33              824.0           757.0 24.75 62.7                52.65
## 34              759.0           827.0 26.50 68.0                35.52
## 35              708.0           757.0 27.50 70.0                34.45
## 36              810.0           790.0 25.75 64.5                43.54
## 37              835.0           821.0 23.00 54.0                33.11
## 38             1023.0           729.0 14.50 20.0                18.26
## 39             1035.0           706.0 19.00 43.0                34.99
## 40             1022.0           698.0 21.00 57.0                33.78
## 41              752.0           715.0  2.50 20.0                35.66
## 42              797.0           683.0 23.00 65.0                33.51
```

```
## 43                 829.0             710.0 13.00 38.0                33.51
## 44                 859.0             797.0 24.00 59.0                27.62
## 45                 879.0             815.0  3.00 20.0                30.97
## 46                 861.0             737.0 17.50 48.0                31.77
## 47                 883.0             679.0 25.50 64.0                37.39
## 48                 904.0             696.0 25.00 61.0                43.01
## 49                 900.0             806.0  0.00 20.0                58.53
## 50                 884.0             792.0 21.50 42.0                52.65
## 51                 866.0             776.0 23.50 60.0                45.69
## 52                 770.0             747.0 21.00 61.0                32.04
## 53                 801.0             778.0  8.00 30.0                36.46
## 54                 912.0             680.0 24.00 62.0                38.59
## 55                 951.0             709.0 20.50 61.5                45.42
## 56                 821.0             818.0 23.00 50.0                19.19
## 57                 904.0             765.0 22.00 40.0                31.50
## 58                 846.0             758.0 22.00 49.0                29.63
## 59                 891.0             672.0 25.00 69.0                26.42
## 60                 869.0             656.0 24.00 65.0                29.50
## 61                 875.0             799.0 19.00 48.0                32.71
## 62                 908.0             829.0 22.50 48.5                39.93
## 63                 881.0             745.0 25.00 63.0                28.29
## 64                 857.0             725.0 23.00 69.0                30.43
## 65                 833.0             790.0 27.00 60.0                37.39
## 66                1041.0             683.0 21.00 51.0                35.39
## 67                1049.0             688.0 18.00 48.0                37.66
## 68                 859.0             827.0 21.00 54.0                40.34
## 69                 721.0             723.0  2.00 20.0                46.36
## 70                 756.0             787.0 26.00 64.0                31.90
## 71                 870.0             768.0  0.00 20.0                44.08
## 72                 794.0             789.0 23.00 58.0                28.16
## 73                 818.0             813.0 25.50 67.0                29.77
## 74                 747.0             778.0 24.00 47.0                41.27
## 75                 875.0             765.0 24.00 67.0                27.89
## 76                 892.0             780.0 23.50 58.5                28.70
## 77                 990.0             789.0 12.00 39.0                32.57
## 78                 864.0             761.0 23.00 63.5                34.18
## 79                1049.9             780.5 16.25 31.0                30.83
## 80                 996.1             789.2 23.50 53.0                30.43
## 81                1049.5             710.1 24.50 57.0                26.42
## 82                1020.9             683.8 21.00 64.0                26.28
## 83                 781.0             841.1 24.00 75.0                36.19
## 84                1028.4             742.7 21.25 46.0                36.32
## 85                1047.6             704.0 23.50 52.5                33.78
## 86                1017.7             681.4 24.50 60.0                30.97
## 87                 964.8             647.1 25.00 55.0                27.09
## 88                 780.6             811.3 26.50 78.0                38.46
## 89                1000.2             667.2  9.50 27.5                37.92
## 90                 999.5             670.5 14.50 36.5                38.19
## 91                 966.8             652.5 14.50 41.5                35.52
## 92                 883.2             652.6 17.00 27.0                32.84
## 93                 780.1             715.3 25.00 78.0                44.48
## 94                 953.3             641.4  0.00 20.0                41.54
## 95                 952.6             644.1  0.00 20.0                41.81
## 96                 920.2             640.9  0.00 20.0                41.01
```

```
## 97              884.0          649.1 27.50 64.0                39.13
## 98              780.3          722.9 25.00 77.0                44.08
## 99              954.2          640.6  0.00 20.0                49.97
## 100             949.9          644.1  2.00 20.0                50.23
## 101             938.9          646.0  0.00 20.0                50.50
## 102             908.9          651.8 27.50 67.0                49.17
## 103             786.2          758.1 29.00 78.0                48.77
```

(That's better!)

Now onward!

Basic Statistics

Let's start by doing some mom-and-apple pie basic statistics and related plots.

Let's look at three variables,

- Cement Amount
- Water Amount
- Compressive Strength

(I also like to arrange things in data frames especially when working with R Markdown Documents)

Remember that these represent our unskilled (or low-skilled) estimates of our parameters.

```
Cement_Stats  = data.frame(n                  = length(  x = exceldata$Cement), # start by loading up th
                           mean               = mean(    x = exceldata$Cement), # data_frame with values
                           standard_deviation = sd(      x = exceldata$Cement),
                           variance           = var(     x = exceldata$Cement),
                           skewness           = skewness(x = exceldata$Cement),
                           kurtosis           = kurtosis(x = exceldata$Cement),
                           row.names          = "Cement Amount") # row names make a label for the frame


Water_Stats  = data.frame(n                  = length(  x = exceldata$Water),
                          mean               = mean(    x = exceldata$Water),
                          standard_deviation = sd(      x = exceldata$Water),
                          variance           = var(     x = exceldata$Water),
                          skewness           = skewness(x = exceldata$Water),
                          kurtosis           = kurtosis(x = exceldata$Water),
                          row.names          = "Water Amount")

Strength_Stats= data.frame(n                  = length(  x = exceldata$Compressive.Strength),
                           mean               = mean(    x = exceldata$Compressive.Strength),
                           standard_deviation = sd(      x = exceldata$Compressive.Strength),
                           variance           = var(     x = exceldata$Compressive.Strength),
                           skewness           = skewness(x = exceldata$Compressive.Strength),
                           kurtosis           = kurtosis(x = exceldata$Compressive.Strength),
                           row.names          = "Compressive Strength")
```

Want 95& confidence intervals? You know the formula…

$CI = t_{\alpha,DF}\frac{s_x}{\sqrt{n}}$

(Need to add a formula? Formulas in R-Markdown notebooks use the classic "hard-core" pre-MS Word era editor LaTex. Don't want to learn LaTek? No worries, neither do I. This webpage will help you poke-and-click a formula and put it in "LaTekeese" Just wrap the code between two $-signs)

The t-statistic can be accessed with the function qt() (like Mathcad and MATLAB)

```
confidence_level = 0.95
alpha            = 1.00 - confidence_level

# the old fashioned way...

Cement_Stats$confidence_limit.95   = qt(p  = 1-alpha/2 ,
                                         df = Cement_Stats$n-1) *
                                      Cement_Stats$standard_deviation /
                                      sqrt(Cement_Stats$n)

Water_Stats$confidence_limit.95    = qt(p  = 1-alpha/2 ,
                                         df = Cement_Stats$n-1) *
                                      Water_Stats$standard_deviation /
                                      sqrt(Water_Stats$n)

Strength_Stats$confidence_limit.95 = qt(p  = 1-alpha/2 ,
                                         df = Strength_Stats$n-1) *
                                      Strength_Stats$standard_deviation /
                                      sqrt(Strength_Stats$n)

print(Cement_Stats)
```

```
##                  n     mean standard_deviation variance  skewness  kurtosis
## Cement Amount 103 229.8942           78.87723 6221.617 0.1409404 -1.691303
##               confidence_limit.95
## Cement Amount            15.41573
```

```
print(Water_Stats)
```

```
##                 n    mean standard_deviation variance  skewness   kurtosis
## Water Amount 103 197.168           20.20816 408.3696 0.2559062 -0.8549319
##              confidence_limit.95
## Water Amount            3.949474
```

```
print(Strength_Stats)
```

```
##                         n     mean standard_deviation variance  skewness
## Compressive Strength 103 36.03942           7.838232 61.43788 0.1866717
##                        kurtosis confidence_limit.95
## Compressive Strength 0.07456379            1.531901
```

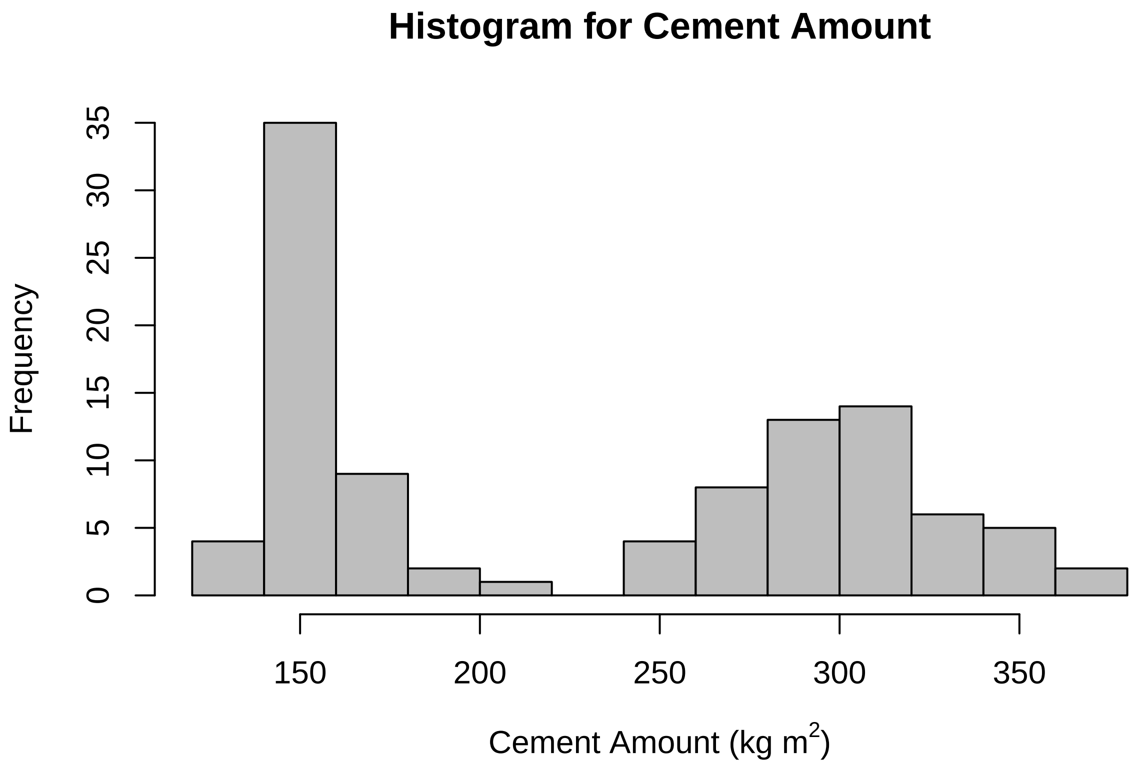
Visualizing our Data

We can also do some basic histogram and box whisker plots.

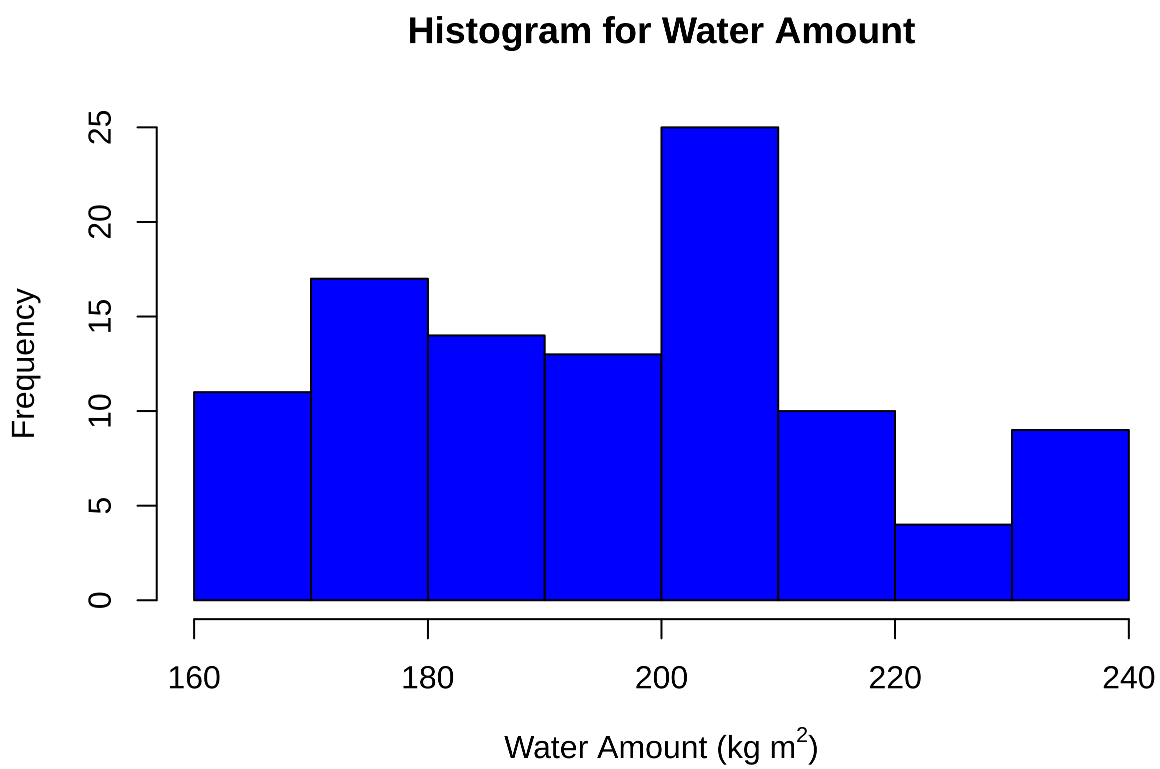
For a histogram, command is "hist()"

```
# plotting histograms

hist(x      = exceldata$Cement,                         # data to process
     breaks = 10,                                       # # of intervals
     main   = "Histogram for Cement Amount",            # main title string
     xlab   = expression('Cement Amount (kg m'^2*")"), # this lets us use superscripts/subscripts
     col    = "grey",                                   # color for shading
```

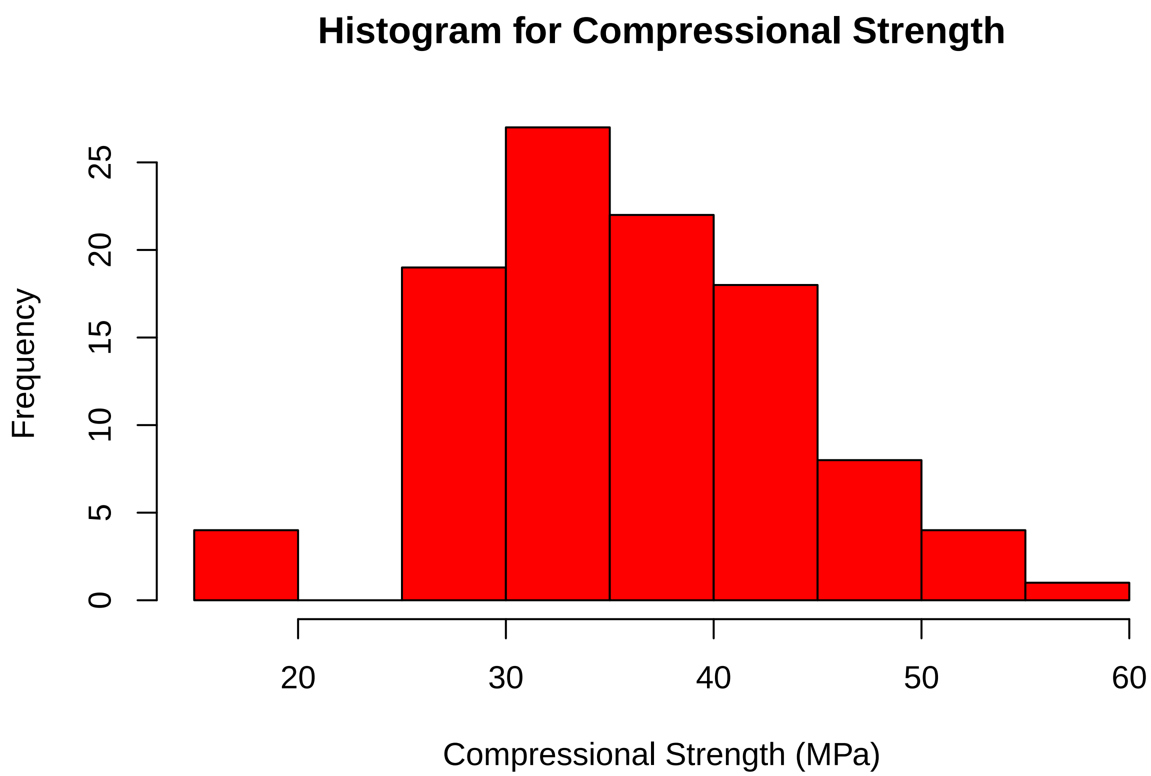
```r
     border = "black")                                          # border for shading
```




```r
hist(x      = exceldata$Water,                                  # data to process
     breaks = 10,                                               # # of intervals
     main   = "Histogram for Water Amount",                     # main title string
     xlab   = expression('Water Amount (kg m'^2*")"),           # x axis label
     col    = "blue",                                           # color for shading
     border = "black")                                          # border for shading
```

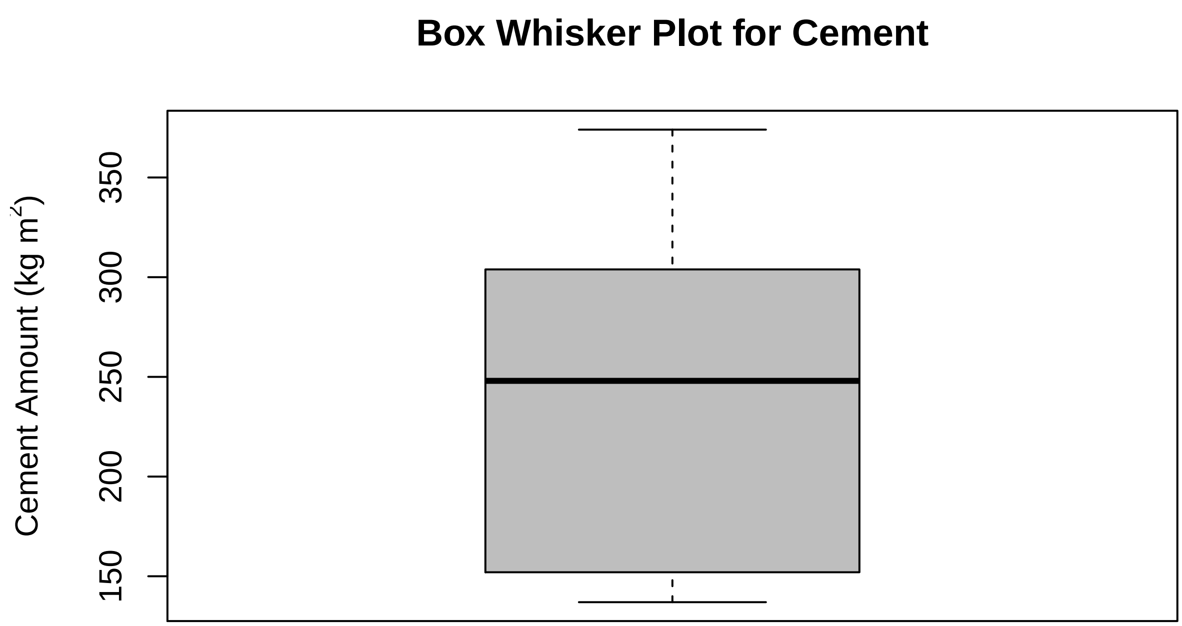



```r
hist(x      = exceldata$Compressive.Strength,          # data to process
     breaks = 10,                                      # # of intervals
     main   = "Histogram for Compressional Strength", # main title string
     xlab   = "Compressional Strength (MPa)",          # x axis label
     col    = "red",                                   # color for shading
     border = "black")                                 # border for shading
```

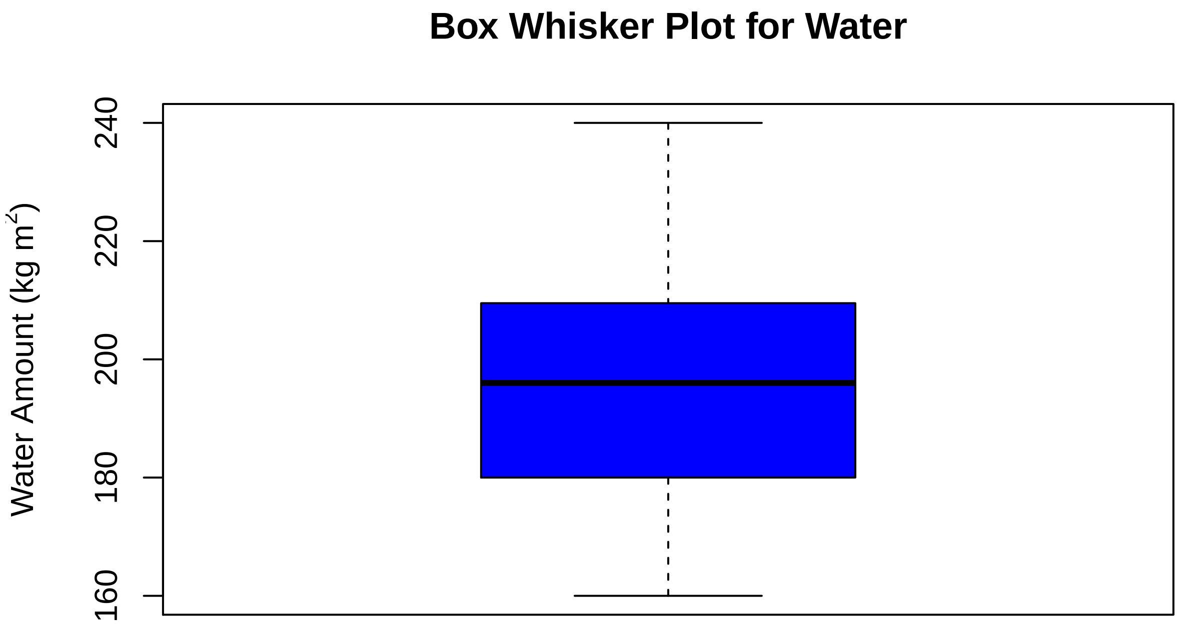


For a box whisker, command is “boxplot()”

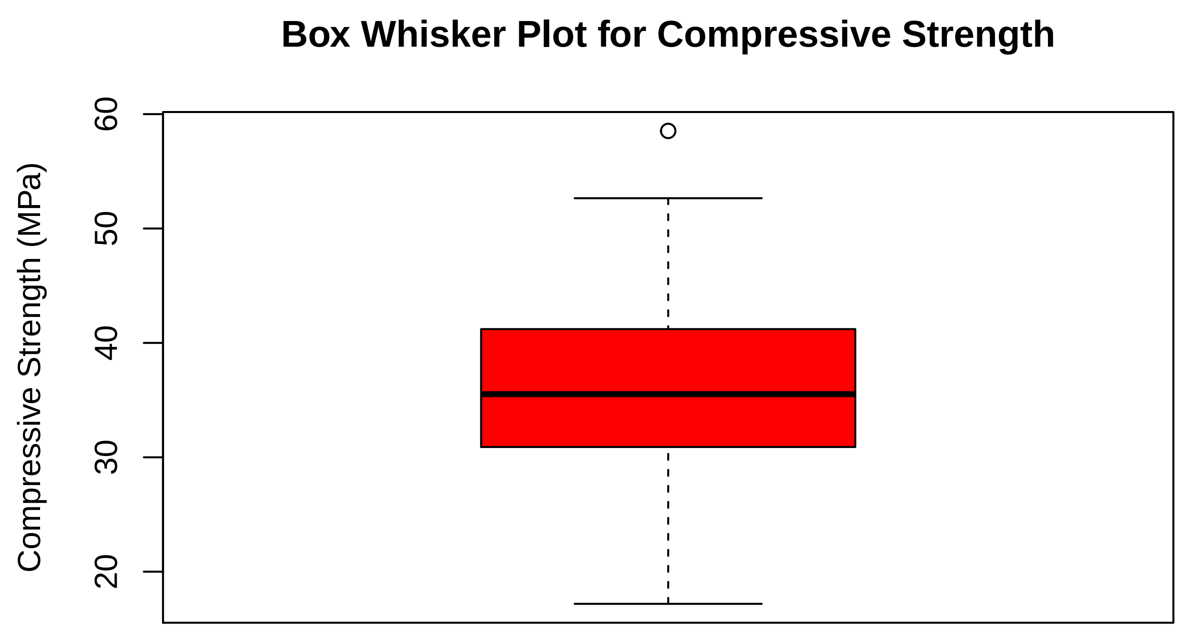
```
# plotting box plots

boxplot(x = exceldata$Cement,                              # data to process
        main = "Box Whisker Plot for Cement",              # main title string
        ylab = expression('Cement Amount (kg m'^2*")"),    # y axis label
         col = "grey",                                     # color for shading
      border = "black")                                    # color for borders
```



```r
boxplot(    x = exceldata$Water,                            # data to process
         main = "Box Whisker Plot for Water",               # main title string
         ylab = expression('Water Amount (kg m'^2*")"),     # y axis label
          col = "blue",                                     # color for shading
       border = "black")                                    # color for borders
```



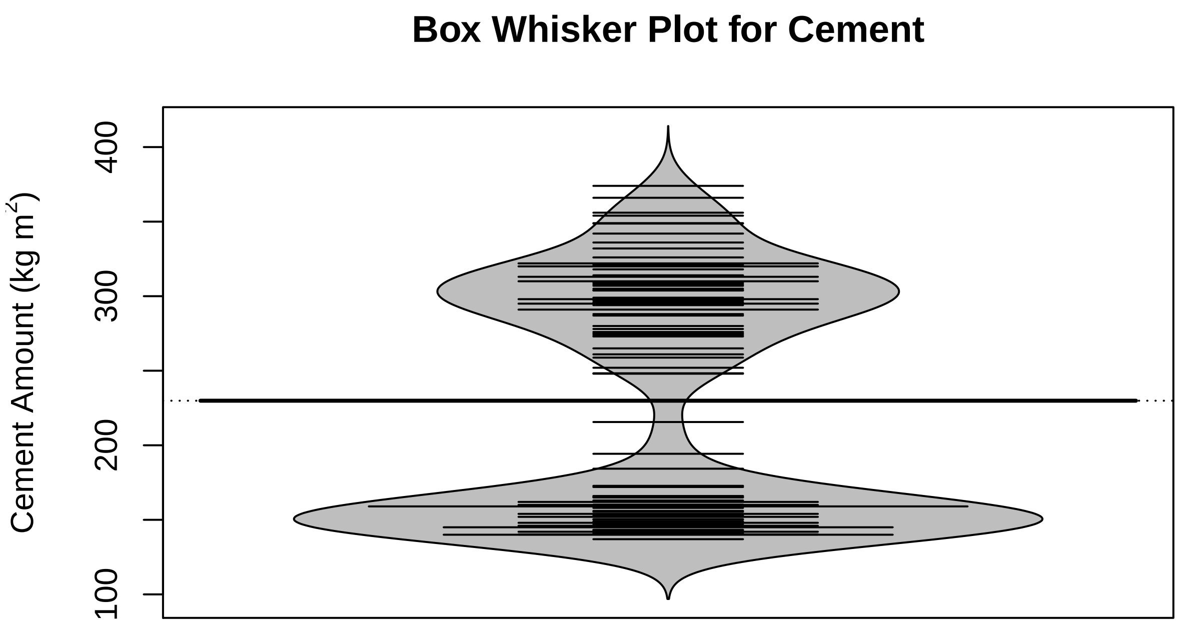

```r
boxplot(    x = exceldata$Compressive.Strength,                # data to process
         main = "Box Whisker Plot for Compressive Strength",   # main title string
         ylab = "Compressive Strength (MPa)",                  # y axis label
          col = "red",                                         # color for shading
       border = "black")                                       # color for borders
```



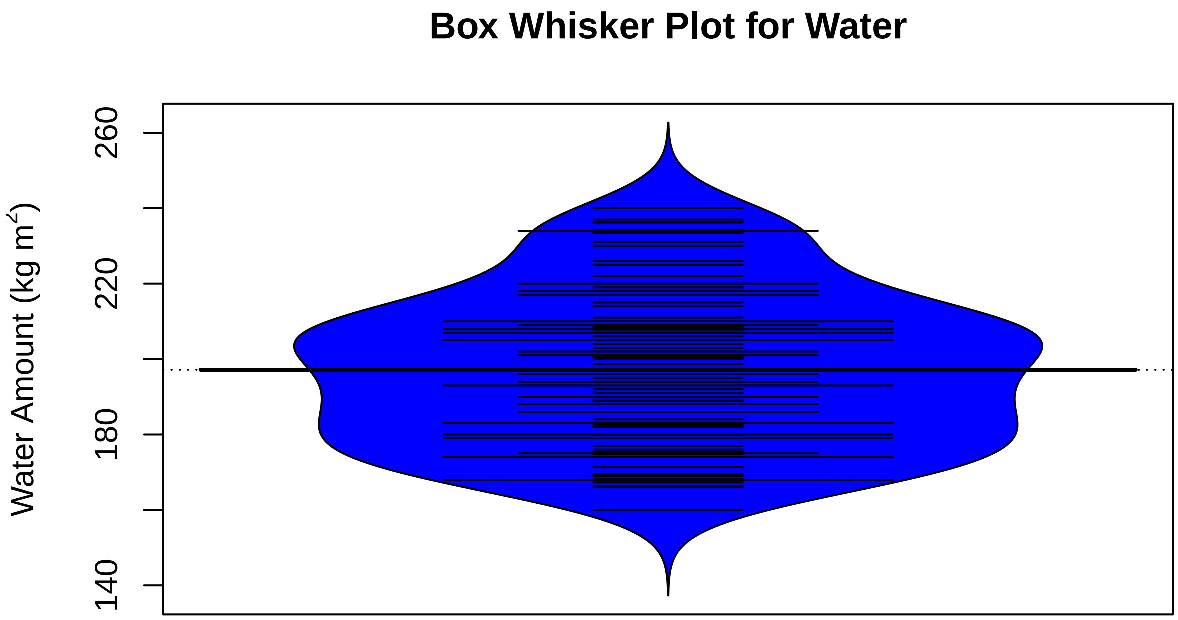
With these we of course see a major limitation in box-whisker plots. We don't, for example, see the bimodal nature of the Concrerte data. It would be nice to see a variant of the box-wisker that gives is the "looking-down" on the histogram advangtate of the box whsiker with the ability to see "nuance" of various features of the histogram.

Luckilly lots people have and a blessed few have invested the time to write code to mae this happen. There are two major kinds of "alterantive" plots out there. The "Violin Plot" (not my favorite of the two) is one but I prefer the beanplot().

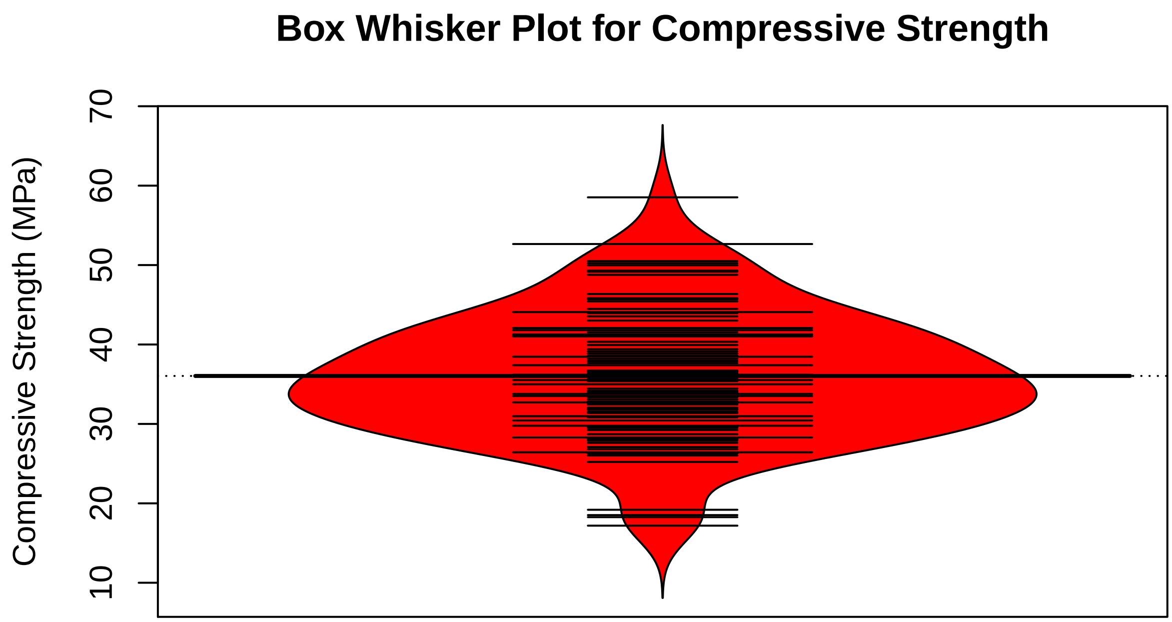
```
# plotting box plots

beanplot(   x = exceldata$Cement,                              # data to process
         main = "Box Whisker Plot for Cement",              # main title string
         ylab = expression('Cement Amount (kg m'^2*")"), # y axis label
          col = "grey",                                     # color for shading
       border = "black")                                    # color for borders
```




```
beanplot(   x = exceldata$Water,                            # data to process
         main = "Box Whisker Plot for Water",               # main title string
         ylab = expression('Water Amount (kg m'^2*")"),   # y axis label
          col = "blue",                                     # color for shading
       border = "black")                                    # color for borders
```




```r
beanplot(  x = exceldata$Compressive.Strength,              # data to process
        main = "Box Whisker Plot for Compressive Strength", # main title string
        ylab = "Compressive Strength (MPa)",                # y axis label
         col = "red",                                       # color for shading
      border = "black")                                     # color for borders
```



So basically the bean plot resembles a hand-drawn distribution but reflected around an axis (and as you saw in our class powerpoints, both halves can represent two different datasets.)

Correlating and then Fitting Cement to Compressive Strength

Let's start by doing a "simple"" plot . In this case since I already know the answer because the spreadsheet also has a table of how well our independent variables correlate against the dependent variables (e.g., Slump, Flow, or in our case Strength). The Cement correlates the best against Compressive Strength (OK, truth be told, it correlates the least badly).

We can acutally do this with a correlate function, cor()…

To grab a value in the table "exceldata" we call the data frame (exceldata) and the variable name (Cement or Water vs Compressive.Strength), separating the frame and variable names by a $ sign.

```
print("Cement vs Compressive Strength Correlation, r")
```

```
## [1] "Cement vs Compressive Strength Correlation, r"
```

```
cor(x = exceldata$Cement,                 # the x-value
    y = exceldata$Compressive.Strength, # the y-value
    method = "pearson"                   # method of correlation
    )
```

```
## [1] 0.4457248
```

or if you like to do everything at once… (Not always the best thing to do)

```
print("Correlation (r) of all variables against Compressive Strength")
```

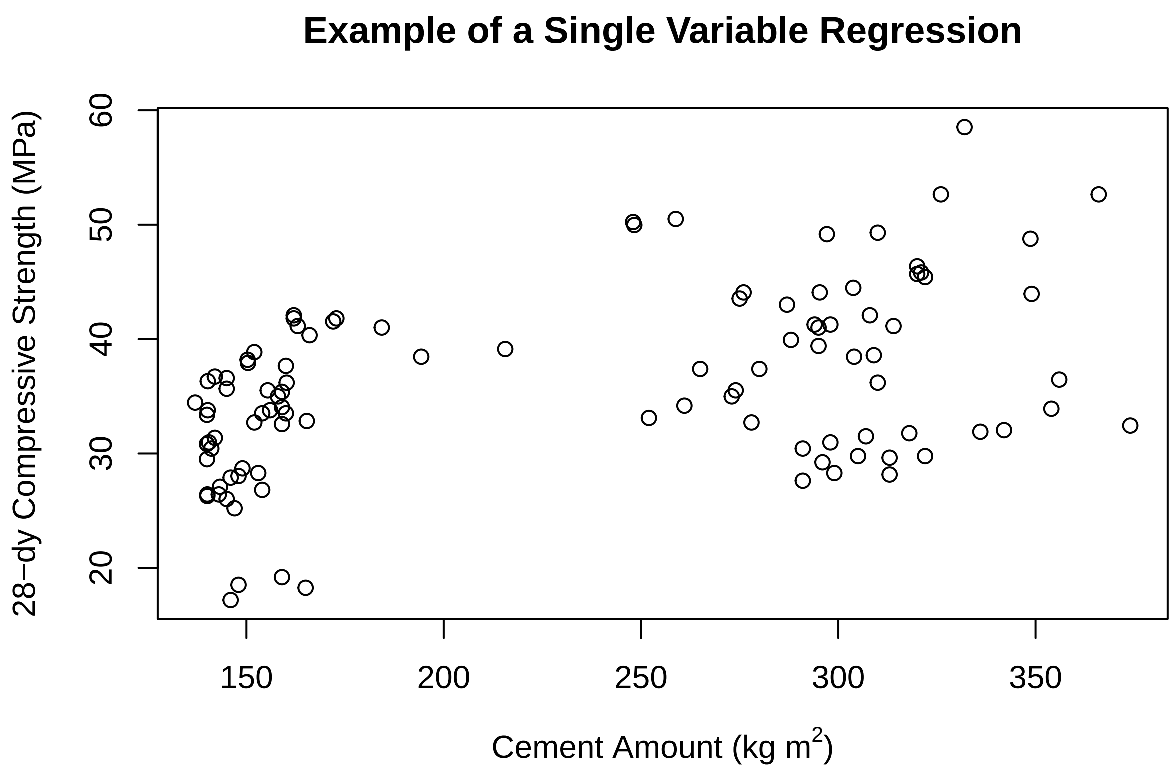
```
## [1] "Correlation (r) of all variables against Compressive Strength"
```

```
cor(x = exceldata,                        # this time X is the whole data frame...
    y = exceldata$Compressive.Strength, # the y-value
    use = "everything",                  # correlate evything
    method = "pearson"
    )
```

```
##                             [,1]
## Sample.Number         0.18627356
## Cement                0.44572481
## Slag                 -0.33158802
## Fly.ash               0.44439260
## Water                -0.25423498
## Superplasticizer     -0.03787084
## Coarse.Aggregates    -0.16068394
## Fine.Aggregates      -0.15448431
## SLUMP                -0.22335810
## FLOW                 -0.12402942
## Compressive.Strength  1.00000000
```

Now to plot the Cement vs Strength as a simople x-y scatter plot.

```
# Now we can plot

plot(x    = exceldata$Cement,                          # x-values (the $ lets us reach into
     y    = exceldata$Compressive.Strength,            # y-values     the data frame)
     main = "Example of a Single Variable Regression", # main title string
     xlab = expression('Cement Amount (kg m'^2*")"),   # x-labels
     ylab = "28-dy Compressive Strength (MPa)"         # y-labels
     )
```



Creating our linear model and "calibrating" it

Not so good a correlation there...

But let's move on and create a regression model from this.

Here we will use the lm() (linear model) function from the MASS package.

For the regression formula

$Strengh = \alpha_0 + \alpha_1 \ concrete$

the "prototype" (formula) for the function is written as ...

"Y ~ X" (with the y-intercept implicit in the formula... you don't put it in but it'll be there when you're done.)

The above syntax is works like this....

Dependant Variable [~ is a function of] Independant Variable [and any other parameter you need gets added with a plus]

If this were a $\hat{y}(x) = \alpha_0 + \alpha_0 \ x^2$, then the prototype for the function would be y~[x^2]

This will hopefully make more sense as we continue!

```
linear_model.S_v_c =  lm(formula = Compressive.Strength ~ Cement, # your formula y ~ x
                         data    = exceldata)                     # the data frame
```

Let's see what we have... This summary command will provide the details of the lm() function's important results

For us we want to see the Y-Intercept [the (Intercept) under "Estimate"] and the slope that goes with our independent value ("Concrete" under "Estimate")

The Standard Error of the Estimate is there (Residual Standard Error) as is the Coefficient of Determination (Multiple R-squared)

We'll talk about a few of the other features when we do the larger multivariate regression

```
summary(object = linear_model.S_v_c)
```

```
##
## Call:
## lm(formula = Compressive.Strength ~ Cement, data = exceldata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1335  -5.3126   0.8321   5.1553  17.9680
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 25.856758   2.150223  12.025  < 2e-16 ***
## Cement       0.044293   0.008851   5.004 2.38e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.051 on 101 degrees of freedom
## Multiple R-squared:  0.1987, Adjusted R-squared:  0.1907
## F-statistic: 25.04 on 1 and 101 DF,  p-value: 2.379e-06
```

In the above output, the asterisk identify the most signiicant independant variables. Here it's trivial even though this is a terrible relationship between cement and strength. Later we will use all of our available independant variables and the use of these asteriks will become more important.

Now let's make a pair of 95% confidence limits

Create a simple linear array to stretch from our min to max dependent values

I tend to overcompensate here and make a BIG array with the sequence (seq) function

Here I am making a variable called "newx" that will represent 501 values of Cement going from 0 to 500 kg m-3.

```
newx <- seq(from       =   0,  # start for a sequence
            to         = 500,  # end for a sequence
            length.out = 501)  # number of elements in a sequence
```

And here we create an array of the confidence limits using this "newx" field

(Note that use can use this function "predict" to do both prediction and confidence limits and also remember that prediction limits should ONLY be used when your original data samples conform to a normal distribution – which often isn't the case.)

```
conf.S_v_c <- predict(object   = linear_model.S_v_c,      # your regression model
                      interval = c("confidence"),         # confidence or prediction intervals
                      level    = 0.95,                    # 1-alpha (here it's a 95% CI)
                      newdata  = data.frame(Cement=newx) # swapping out of new indep variables
                      )
```

The resulting array, conf, will have three columns: the original linear regression using our newx as the independent value and the + & - CI's

```
data.frame(conf.S_v_c)
```

```
##          fit      lwr      upr
## 1   25.85676 21.59129 30.12222
## 2   25.90105 21.65220 30.14990
## 3   25.94534 21.71310 30.17759
## 4   25.98964 21.77399 30.20528
## 5   26.03393 21.83487 30.23299
## 6   26.07822 21.89575 30.26070
## 7   26.12251 21.95661 30.28842
## 8   26.16681 22.01747 30.31614
## 9   26.21110 22.07832 30.34388
## 10  26.25539 22.13916 30.37162
## 11  26.29969 22.20000 30.39937
## 12  26.34398 22.26082 30.42713
## 13  26.38827 22.32164 30.45490
## 14  26.43256 22.38245 30.48268
## 15  26.47686 22.44325 30.51047
## 16  26.52115 22.50404 30.53826
## 17  26.56544 22.56482 30.56607
## 18  26.60974 22.62559 30.59388
## 19  26.65403 22.68635 30.62170
## 20  26.69832 22.74711 30.64953
## 21  26.74261 22.80785 30.67738
## 22  26.78691 22.86859 30.70523
## 23  26.83120 22.92931 30.73309
## 24  26.87549 22.99003 30.76096
## 25  26.91979 23.05073 30.78884
## 26  26.96408 23.11142 30.81673
## 27  27.00837 23.17211 30.84463
## 28  27.05266 23.23278 30.87254
## 29  27.09696 23.29345 30.90047
## 30  27.14125 23.35410 30.92840
## 31  27.18554 23.41474 30.95634
## 32  27.22983 23.47537 30.98430
## 33  27.27413 23.53599 31.01226
## 34  27.31842 23.59660 31.04024
## 35  27.36271 23.65720 31.06823
## 36  27.40701 23.71779 31.09623
## 37  27.45130 23.77836 31.12424
## 38  27.49559 23.83892 31.15226
## 39  27.53988 23.89947 31.18030
## 40  27.58418 23.96001 31.20834
## 41  27.62847 24.02054 31.23640
## 42  27.67276 24.08105 31.26447
## 43  27.71706 24.14155 31.29256
## 44  27.76135 24.20204 31.32066
## 45  27.80564 24.26252 31.34877
## 46  27.84993 24.32298 31.37689
## 47  27.89423 24.38343 31.40503
## 48  27.93852 24.44386 31.43318
## 49  27.98281 24.50428 31.46134
## 50  28.02711 24.56469 31.48952
## 51  28.07140 24.62508 31.51771
## 52  28.11569 24.68546 31.54592
## 53  28.15998 24.74583 31.57414
```

```
## 54  28.20428 24.80618 31.60238
## 55  28.24857 24.86651 31.63063
## 56  28.29286 24.92683 31.65889
## 57  28.33716 24.98714 31.68717
## 58  28.38145 25.04743 31.71547
## 59  28.42574 25.10770 31.74378
## 60  28.47003 25.16796 31.77211
## 61  28.51433 25.22820 31.80046
## 62  28.55862 25.28842 31.82882
## 63  28.60291 25.34863 31.85720
## 64  28.64720 25.40882 31.88559
## 65  28.69150 25.46899 31.91400
## 66  28.73579 25.52915 31.94243
## 67  28.78008 25.58929 31.97088
## 68  28.82438 25.64941 31.99935
## 69  28.86867 25.70951 32.02783
## 70  28.91296 25.76959 32.05633
## 71  28.95725 25.82966 32.08485
## 72  29.00155 25.88970 32.11339
## 73  29.04584 25.94973 32.14195
## 74  29.09013 26.00973 32.17053
## 75  29.13443 26.06972 32.19913
## 76  29.17872 26.12968 32.22775
## 77  29.22301 26.18963 32.25639
## 78  29.26730 26.24955 32.28506
## 79  29.31160 26.30946 32.31374
## 80  29.35589 26.36934 32.34244
## 81  29.40018 26.42920 32.37117
## 82  29.44448 26.48903 32.39992
## 83  29.48877 26.54885 32.42869
## 84  29.53306 26.60864 32.45749
## 85  29.57735 26.66840 32.48630
## 86  29.62165 26.72815 32.51515
## 87  29.66594 26.78787 32.54401
## 88  29.71023 26.84756 32.57290
## 89  29.75453 26.90723 32.60182
## 90  29.79882 26.96688 32.63076
## 91  29.84311 27.02650 32.65973
## 92  29.88740 27.08609 32.68872
## 93  29.93170 27.14565 32.71774
## 94  29.97599 27.20519 32.74678
## 95  30.02028 27.26471 32.77586
## 96  30.06457 27.32419 32.80496
## 97  30.10887 27.38364 32.83409
## 98  30.15316 27.44307 32.86325
## 99  30.19745 27.50247 32.89244
## 100 30.24175 27.56183 32.92166
## 101 30.28604 27.62117 32.95091
## 102 30.33033 27.68047 32.98019
## 103 30.37462 27.73975 33.00950
## 104 30.41892 27.79899 33.03885
## 105 30.46321 27.85820 33.06822
## 106 30.50750 27.91738 33.09763
## 107 30.55180 27.97652 33.12707
```

```
## 108 30.59609 28.03563 33.15655
## 109 30.64038 28.09470 33.18606
## 110 30.68467 28.15374 33.21561
## 111 30.72897 28.21274 33.24520
## 112 30.77326 28.27170 33.27482
## 113 30.81755 28.33063 33.30448
## 114 30.86185 28.38952 33.33417
## 115 30.90614 28.44837 33.36391
## 116 30.95043 28.50718 33.39368
## 117 30.99472 28.56595 33.42350
## 118 31.03902 28.62468 33.45335
## 119 31.08331 28.68337 33.48325
## 120 31.12760 28.74202 33.51319
## 121 31.17190 28.80062 33.54317
## 122 31.21619 28.85918 33.57320
## 123 31.26048 28.91769 33.60327
## 124 31.30477 28.97616 33.63339
## 125 31.34907 29.03458 33.66355
## 126 31.39336 29.09296 33.69376
## 127 31.43765 29.15129 33.72402
## 128 31.48195 29.20956 33.75433
## 129 31.52624 29.26779 33.78468
## 130 31.57053 29.32597 33.81509
## 131 31.61482 29.38410 33.84555
## 132 31.65912 29.44217 33.87606
## 133 31.70341 29.50019 33.90663
## 134 31.74770 29.55815 33.93725
## 135 31.79199 29.61606 33.96793
## 136 31.83629 29.67391 33.99866
## 137 31.88058 29.73170 34.02946
## 138 31.92487 29.78944 34.06031
## 139 31.96917 29.84711 34.09122
## 140 32.01346 29.90473 34.12219
## 141 32.05775 29.96228 34.15323
## 142 32.10204 30.01976 34.18432
## 143 32.14634 30.07719 34.21549
## 144 32.19063 30.13454 34.24672
## 145 32.23492 30.19183 34.27801
## 146 32.27922 30.24905 34.30938
## 147 32.32351 30.30620 34.34082
## 148 32.36780 30.36328 34.37232
## 149 32.41209 30.42029 34.40390
## 150 32.45639 30.47722 34.43556
## 151 32.50068 30.53407 34.46728
## 152 32.54497 30.59085 34.49909
## 153 32.58927 30.64755 34.53098
## 154 32.63356 30.70418 34.56294
## 155 32.67785 30.76071 34.59499
## 156 32.72214 30.81717 34.62712
## 157 32.76644 30.87354 34.65933
## 158 32.81073 30.92983 34.69163
## 159 32.85502 30.98602 34.72402
## 160 32.89932 31.04213 34.75650
## 161 32.94361 31.09815 34.78907
```

```
## 162 32.98790 31.15407 34.82173
## 163 33.03219 31.20990 34.85449
## 164 33.07649 31.26563 34.88734
## 165 33.12078 31.32126 34.92030
## 166 33.16507 31.37679 34.95335
## 167 33.20936 31.43222 34.98651
## 168 33.25366 31.48755 35.01977
## 169 33.29795 31.54277 35.05313
## 170 33.34224 31.59788 35.08661
## 171 33.38654 31.65288 35.12019
## 172 33.43083 31.70777 35.15389
## 173 33.47512 31.76254 35.18770
## 174 33.51941 31.81720 35.22163
## 175 33.56371 31.87173 35.25568
## 176 33.60800 31.92615 35.28985
## 177 33.65229 31.98044 35.32414
## 178 33.69659 32.03461 35.35856
## 179 33.74088 32.08865 35.39310
## 180 33.78517 32.14256 35.42778
## 181 33.82946 32.19634 35.46258
## 182 33.87376 32.24999 35.49753
## 183 33.91805 32.30350 35.53260
## 184 33.96234 32.35686 35.56782
## 185 34.00664 32.41009 35.60318
## 186 34.05093 32.46317 35.63868
## 187 34.09522 32.51611 35.67433
## 188 34.13951 32.56890 35.71013
## 189 34.18381 32.62154 35.74608
## 190 34.22810 32.67402 35.78218
## 191 34.27239 32.72635 35.81844
## 192 34.31669 32.77852 35.85485
## 193 34.36098 32.83052 35.89143
## 194 34.40527 32.88237 35.92817
## 195 34.44956 32.93405 35.96508
## 196 34.49386 32.98556 36.00216
## 197 34.53815 33.03690 36.03940
## 198 34.58244 33.08806 36.07682
## 199 34.62673 33.13905 36.11441
## 200 34.67103 33.18987 36.15219
## 201 34.71532 33.24050 36.19014
## 202 34.75961 33.29095 36.22827
## 203 34.80391 33.34122 36.26659
## 204 34.84820 33.39130 36.30510
## 205 34.89249 33.44119 36.34380
## 206 34.93678 33.49088 36.38269
## 207 34.98108 33.54039 36.42177
## 208 35.02537 33.58969 36.46105
## 209 35.06966 33.63880 36.50052
## 210 35.11396 33.68771 36.54020
## 211 35.15825 33.73642 36.58007
## 212 35.20254 33.78493 36.62016
## 213 35.24683 33.83323 36.66044
## 214 35.29113 33.88132 36.70094
## 215 35.33542 33.92920 36.74164
```

```
## 216 35.37971 33.97687 36.78255
## 217 35.42401 34.02433 36.82368
## 218 35.46830 34.07158 36.86502
## 219 35.51259 34.11861 36.90657
## 220 35.55688 34.16543 36.94834
## 221 35.60118 34.21202 36.99033
## 222 35.64547 34.25840 37.03254
## 223 35.68976 34.30456 37.07496
## 224 35.73406 34.35050 37.11761
## 225 35.77835 34.39622 37.16048
## 226 35.82264 34.44172 37.20356
## 227 35.86693 34.48699 37.24688
## 228 35.91123 34.53204 37.29041
## 229 35.95552 34.57687 37.33417
## 230 35.99981 34.62148 37.37815
## 231 36.04410 34.66586 37.42235
## 232 36.08840 34.71001 37.46678
## 233 36.13269 34.75395 37.51143
## 234 36.17698 34.79766 37.55631
## 235 36.22128 34.84114 37.60141
## 236 36.26557 34.88441 37.64673
## 237 36.30986 34.92745 37.69227
## 238 36.35415 34.97027 37.73804
## 239 36.39845 35.01287 37.78402
## 240 36.44274 35.05525 37.83023
## 241 36.48703 35.09741 37.87666
## 242 36.53133 35.13935 37.92330
## 243 36.57562 35.18108 37.97016
## 244 36.61991 35.22258 38.01724
## 245 36.66420 35.26388 38.06453
## 246 36.70850 35.30496 38.11203
## 247 36.75279 35.34583 38.15975
## 248 36.79708 35.38649 38.20768
## 249 36.84138 35.42694 38.25581
## 250 36.88567 35.46718 38.30416
## 251 36.92996 35.50722 38.35270
## 252 36.97425 35.54705 38.40146
## 253 37.01855 35.58669 38.45041
## 254 37.06284 35.62612 38.49956
## 255 37.10713 35.66535 38.54891
## 256 37.15143 35.70440 38.59845
## 257 37.19572 35.74324 38.64819
## 258 37.24001 35.78190 38.69812
## 259 37.28430 35.82037 38.74824
## 260 37.32860 35.85865 38.79855
## 261 37.37289 35.89674 38.84904
## 262 37.41718 35.93466 38.89971
## 263 37.46147 35.97239 38.95056
## 264 37.50577 36.00995 39.00159
## 265 37.55006 36.04733 39.05279
## 266 37.59435 36.08454 39.10417
## 267 37.63865 36.12158 39.15571
## 268 37.68294 36.15845 39.20743
## 269 37.72723 36.19516 39.25931
```

```
## 270 37.77152 36.23170 39.31135
## 271 37.81582 36.26808 39.36355
## 272 37.86011 36.30431 39.41591
## 273 37.90440 36.34038 39.46843
## 274 37.94870 36.37630 39.52110
## 275 37.99299 36.41206 39.57392
## 276 38.03728 36.44768 39.62688
## 277 38.08157 36.48315 39.68000
## 278 38.12587 36.51848 39.73325
## 279 38.17016 36.55367 39.78665
## 280 38.21445 36.58872 39.84019
## 281 38.25875 36.62363 39.89386
## 282 38.30304 36.65841 39.94767
## 283 38.34733 36.69305 40.00161
## 284 38.39162 36.72757 40.05568
## 285 38.43592 36.76196 40.10987
## 286 38.48021 36.79623 40.16419
## 287 38.52450 36.83037 40.21863
## 288 38.56880 36.86439 40.27320
## 289 38.61309 36.89830 40.32788
## 290 38.65738 36.93208 40.38268
## 291 38.70167 36.96576 40.43759
## 292 38.74597 36.99932 40.49261
## 293 38.79026 37.03277 40.54775
## 294 38.83455 37.06612 40.60299
## 295 38.87884 37.09935 40.65833
## 296 38.92314 37.13249 40.71379
## 297 38.96743 37.16552 40.76934
## 298 39.01172 37.19845 40.82499
## 299 39.05602 37.23129 40.88074
## 300 39.10031 37.26403 40.93659
## 301 39.14460 37.29667 40.99254
## 302 39.18889 37.32922 41.04857
## 303 39.23319 37.36168 41.10470
## 304 39.27748 37.39405 41.16091
## 305 39.32177 37.42633 41.21722
## 306 39.36607 37.45853 41.27361
## 307 39.41036 37.49064 41.33008
## 308 39.45465 37.52267 41.38664
## 309 39.49894 37.55461 41.44327
## 310 39.54324 37.58648 41.49999
## 311 39.58753 37.61827 41.55679
## 312 39.63182 37.64999 41.61366
## 313 39.67612 37.68162 41.67061
## 314 39.72041 37.71319 41.72763
## 315 39.76470 37.74468 41.78472
## 316 39.80899 37.77610 41.84189
## 317 39.85329 37.80745 41.89912
## 318 39.89758 37.83873 41.95643
## 319 39.94187 37.86995 42.01380
## 320 39.98617 37.90110 42.07123
## 321 40.03046 37.93218 42.12873
## 322 40.07475 37.96320 42.18630
## 323 40.11904 37.99416 42.24392
```

```
## 324 40.16334 38.02506 42.30161
## 325 40.20763 38.05590 42.35936
## 326 40.25192 38.08668 42.41716
## 327 40.29621 38.11740 42.47503
## 328 40.34051 38.14807 42.53295
## 329 40.38480 38.17868 42.59092
## 330 40.42909 38.20923 42.64895
## 331 40.47339 38.23973 42.70704
## 332 40.51768 38.27018 42.76517
## 333 40.56197 38.30058 42.82336
## 334 40.60626 38.33093 42.88160
## 335 40.65056 38.36123 42.93989
## 336 40.69485 38.39147 42.99823
## 337 40.73914 38.42167 43.05661
## 338 40.78344 38.45183 43.11504
## 339 40.82773 38.48193 43.17352
## 340 40.87202 38.51200 43.23205
## 341 40.91631 38.54201 43.29061
## 342 40.96061 38.57199 43.34923
## 343 41.00490 38.60192 43.40788
## 344 41.04919 38.63181 43.46658
## 345 41.09349 38.66165 43.52532
## 346 41.13778 38.69146 43.58410
## 347 41.18207 38.72122 43.64292
## 348 41.22636 38.75095 43.70178
## 349 41.27066 38.78064 43.76067
## 350 41.31495 38.81029 43.81961
## 351 41.35924 38.83990 43.87858
## 352 41.40354 38.86948 43.93759
## 353 41.44783 38.89902 43.99664
## 354 41.49212 38.92852 44.05572
## 355 41.53641 38.95800 44.11483
## 356 41.58071 38.98743 44.17398
## 357 41.62500 39.01683 44.23316
## 358 41.66929 39.04620 44.29238
## 359 41.71358 39.07554 44.35163
## 360 41.75788 39.10485 44.41091
## 361 41.80217 39.13412 44.47022
## 362 41.84646 39.16336 44.52956
## 363 41.89076 39.19257 44.58894
## 364 41.93505 39.22176 44.64834
## 365 41.97934 39.25091 44.70777
## 366 42.02363 39.28004 44.76723
## 367 42.06793 39.30913 44.82672
## 368 42.11222 39.33820 44.88624
## 369 42.15651 39.36724 44.94578
## 370 42.20081 39.39626 45.00536
## 371 42.24510 39.42524 45.06495
## 372 42.28939 39.45420 45.12458
## 373 42.33368 39.48314 45.18423
## 374 42.37798 39.51205 45.24390
## 375 42.42227 39.54094 45.30360
## 376 42.46656 39.56980 45.36333
## 377 42.51086 39.59863 45.42308
```

```
## 378 42.55515 39.62745 45.48285
## 379 42.59944 39.65624 45.54265
## 380 42.64373 39.68500 45.60247
## 381 42.68803 39.71375 45.66231
## 382 42.73232 39.74247 45.72217
## 383 42.77661 39.77117 45.78206
## 384 42.82091 39.79985 45.84196
## 385 42.86520 39.82850 45.90189
## 386 42.90949 39.85714 45.96184
## 387 42.95378 39.88576 46.02181
## 388 42.99808 39.91435 46.08180
## 389 43.04237 39.94293 46.14181
## 390 43.08666 39.97148 46.20184
## 391 43.13095 40.00002 46.26189
## 392 43.17525 40.02854 46.32196
## 393 43.21954 40.05704 46.38205
## 394 43.26383 40.08552 46.44215
## 395 43.30813 40.11398 46.50228
## 396 43.35242 40.14242 46.56242
## 397 43.39671 40.17085 46.62258
## 398 43.44100 40.19926 46.68275
## 399 43.48530 40.22765 46.74295
## 400 43.52959 40.25602 46.80316
## 401 43.57388 40.28438 46.86339
## 402 43.61818 40.31272 46.92363
## 403 43.66247 40.34105 46.98389
## 404 43.70676 40.36936 47.04417
## 405 43.75105 40.39765 47.10446
## 406 43.79535 40.42593 47.16477
## 407 43.83964 40.45419 47.22509
## 408 43.88393 40.48244 47.28543
## 409 43.92823 40.51067 47.34578
## 410 43.97252 40.53889 47.40615
## 411 44.01681 40.56709 47.46653
## 412 44.06110 40.59528 47.52693
## 413 44.10540 40.62345 47.58734
## 414 44.14969 40.65162 47.64776
## 415 44.19398 40.67976 47.70820
## 416 44.23828 40.70790 47.76865
## 417 44.28257 40.73602 47.82912
## 418 44.32686 40.76413 47.88960
## 419 44.37115 40.79222 47.95009
## 420 44.41545 40.82030 48.01059
## 421 44.45974 40.84837 48.07111
## 422 44.50403 40.87643 48.13164
## 423 44.54832 40.90447 48.19218
## 424 44.59262 40.93251 48.25273
## 425 44.63691 40.96053 48.31329
## 426 44.68120 40.98854 48.37387
## 427 44.72550 41.01653 48.43446
## 428 44.76979 41.04452 48.49506
## 429 44.81408 41.07249 48.55567
## 430 44.85837 41.10046 48.61629
## 431 44.90267 41.12841 48.67693
```

```
## 432 44.94696 41.15635 48.73757
## 433 44.99125 41.18428 48.79823
## 434 45.03555 41.21220 48.85889
## 435 45.07984 41.24011 48.91957
## 436 45.12413 41.26801 48.98026
## 437 45.16842 41.29590 49.04095
## 438 45.21272 41.32378 49.10166
## 439 45.25701 41.35164 49.16238
## 440 45.30130 41.37950 49.22310
## 441 45.34560 41.40735 49.28384
## 442 45.38989 41.43519 49.34458
## 443 45.43418 41.46302 49.40534
## 444 45.47847 41.49084 49.46611
## 445 45.52277 41.51865 49.52688
## 446 45.56706 41.54646 49.58766
## 447 45.61135 41.57425 49.64845
## 448 45.65565 41.60203 49.70926
## 449 45.69994 41.62981 49.77007
## 450 45.74423 41.65758 49.83088
## 451 45.78852 41.68534 49.89171
## 452 45.83282 41.71309 49.95255
## 453 45.87711 41.74083 50.01339
## 454 45.92140 41.76856 50.07424
## 455 45.96569 41.79629 50.13510
## 456 46.00999 41.82400 50.19597
## 457 46.05428 41.85171 50.25685
## 458 46.09857 41.87941 50.31773
## 459 46.14287 41.90711 50.37863
## 460 46.18716 41.93479 50.43953
## 461 46.23145 41.96247 50.50043
## 462 46.27574 41.99014 50.56135
## 463 46.32004 42.01780 50.62227
## 464 46.36433 42.04546 50.68320
## 465 46.40862 42.07311 50.74414
## 466 46.45292 42.10075 50.80508
## 467 46.49721 42.12838 50.86603
## 468 46.54150 42.15601 50.92699
## 469 46.58579 42.18363 50.98796
## 470 46.63009 42.21125 51.04893
## 471 46.67438 42.23885 51.10991
## 472 46.71867 42.26645 51.17089
## 473 46.76297 42.29404 51.23189
## 474 46.80726 42.32163 51.29289
## 475 46.85155 42.34921 51.35389
## 476 46.89584 42.37679 51.41490
## 477 46.94014 42.40435 51.47592
## 478 46.98443 42.43191 51.53694
## 479 47.02872 42.45947 51.59798
## 480 47.07302 42.48702 51.65901
## 481 47.11731 42.51456 51.72006
## 482 47.16160 42.54210 51.78110
## 483 47.20589 42.56963 51.84216
## 484 47.25019 42.59715 51.90322
## 485 47.29448 42.62467 51.96429
```

```
## 486 47.33877 42.65219 52.02536
## 487 47.38306 42.67969 52.08644
## 488 47.42736 42.70720 52.14752
## 489 47.47165 42.73469 52.20861
## 490 47.51594 42.76218 52.26970
## 491 47.56024 42.78967 52.33080
## 492 47.60453 42.81715 52.39191
## 493 47.64882 42.84462 52.45302
## 494 47.69311 42.87209 52.51414
## 495 47.73741 42.89956 52.57526
## 496 47.78170 42.92702 52.63638
## 497 47.82599 42.95447 52.69752
## 498 47.87029 42.98192 52.75865
## 499 47.91458 43.00936 52.81979
## 500 47.95887 43.03680 52.88094
## 501 48.00316 43.06424 52.94209
```

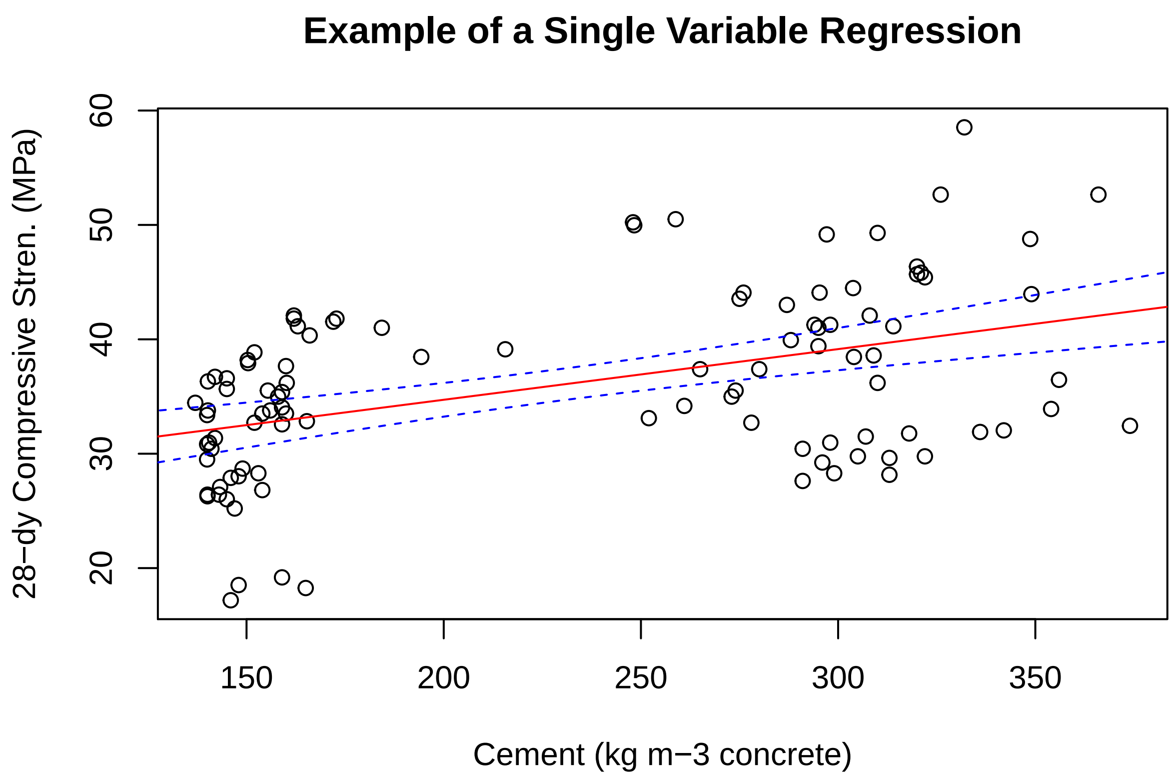
So let's replot our relationship…

…our regression

and finally our lower than upper confidence limits…

(indexing on those confidence limits in "conf.S_v_c"" start at zero, by the way… so the counter is index 0, the regression line value is index 1, and the CI values are 2 & 3)

```r
# Now we can plot

plot(x    = exceldata$Cement,                         # x-values (the $ lets us reach into
     y    = exceldata$Compressive.Strength,           # y-values     the data frame)
     main = "Example of a Single Variable Regression", # main title string
     xlab = "Cement (kg m-3 concrete)",               # x-labels
     ylab = "28-dy Compressive Stren. (MPa)"          # y-labels
     )

# And here we can plot the regression line

abline(reg = linear_model.S_v_c,  # put the regression model information here
       col = "red"                # color it red
       )

# And here we create an array of the confidence limits using this "newx" field

lines(x   = newx,            # the x data
      y   = conf.S_v_c[,2],  # the y data (the low-end confidence limit)
      col = "blue",          # make the line blue
      lty = 2)               # use a dashed line

lines(x   = newx,            # the x data
      y   = conf.S_v_c[,3],  # the y data (the high-end confidence limit)
      col = "blue",          # make the line blue
      lty = 2)               # use a dashed line
```



Looks pretty good! (well the graph does.. not necessarily the quality of the regression)

And now we're going to do something about that!

We're now going to use not just one independent variable… but all 7 of them!

The good news is that it follows the same form as the simple linear regression. This time we string along all of our independent variables with in our formula prototype.

```
linear_model.S_v_all <- lm(data    = exceldata,                           # your data frame
                           formula = Compressive.Strength ~ Cement +   # your formula
                                                            Slag +
                                                            Fly.ash +
                                                            Water +
                                                            Superplasticizer +
                                                            Fine.Aggregates +
                                                            Coarse.Aggregates
                           )
```

And here are these results…

```
summary(object = linear_model.S_v_all)
```

```
##
## Call:
## lm(formula = Compressive.Strength ~ Cement + Slag + Fly.ash +
##     Water + Superplasticizer + Fine.Aggregates + Coarse.Aggregates,
##     data = exceldata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8411 -1.7063 -0.2831  1.2986  7.9424
```

```
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       139.78150   71.10128   1.966  0.05222 .
## Cement              0.06141    0.02282   2.691  0.00842 **
## Slag               -0.02971    0.03176  -0.935  0.35200
## Fly.ash             0.05053    0.02316   2.182  0.03159 *
## Water              -0.23270    0.07166  -3.247  0.00161 **
## Superplasticizer    0.10315    0.13459   0.766  0.44532
## Fine.Aggregates    -0.03908    0.02882  -1.356  0.17833
## Coarse.Aggregates  -0.05562    0.02744  -2.027  0.04546 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.609 on 95 degrees of freedom
## Multiple R-squared:  0.8968, Adjusted R-squared:  0.8892
## F-statistic:   118 on 7 and 95 DF,  p-value: < 2.2e-16
```

Our regression coefficients are still here under the "Estimate" column as are our Standard Error of our Estimate and our Coeff of Determination.

Also we can now take a good look at those asterisks at the end of line with the parameter coefficients. These can explain which independent variables do the heaviest lifting in our regression. The more asterisks, the more important the dependent variable is to the larger multivariate regression. Here, we can see that the Cement and Water are doing most of the "work" in fitting our suite of independent variables to our dependent variable of Compressive Strength.

Finally there is the P parameter for which the smaller it is, the better we can say that the relationship that we've made with our regression represents our dependent variable.

Now… on to looking at our results.

Here is where viewing the results of the regression is tricky.

We have 7 independent variables but we'd like to see the impact of the fit if all 7 variables on our strength

When I do this I like to plot the true y value against my regression y(x1,x2,x3,..)

So to do this I will take the fitted values of y and plot them against the original values of y

Getting the fitted values is easy.

I'm using the fitted function but you can also use the predict function from earlier with the fitted function.

```
fitted.S_v_all <- fitted(object = linear_model.S_v_all)
```

The other thing is to now do a second regression (ok a third in this entire demo…)

Here we will just regress y(x1,x2…) against y.

```
linear_model.S_v_Sofall <- lm(formula = fitted.S_v_all ~ exceldata$Compressive.Strength)

summary(object = linear_model.S_v_Sofall)
```

```
##
## Call:
## lm(formula = fitted.S_v_all ~ exceldata$Compressive.Strength)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7870 -1.6428  0.2609  1.6482  4.9775
```

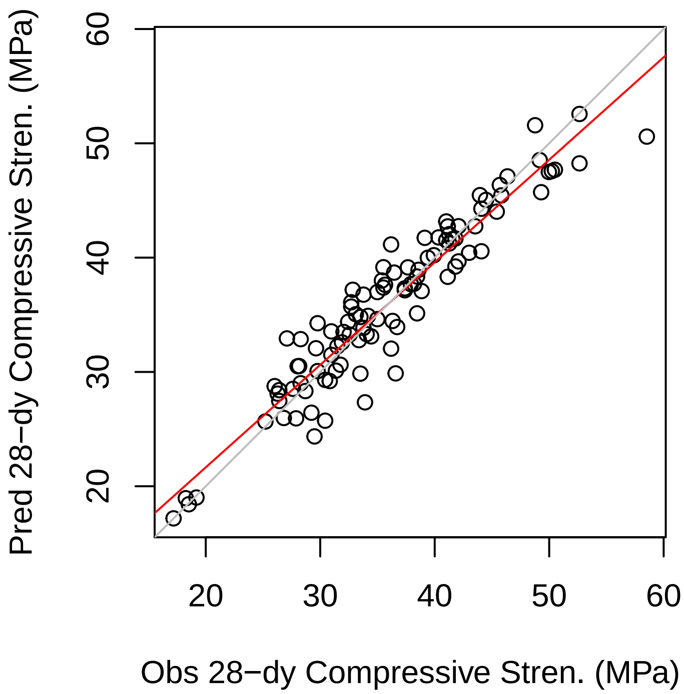
```
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   3.71791    1.11603   3.331  0.00121 **
## exceldata$Compressive.Strength  0.89684    0.03027  29.632  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.396 on 101 degrees of freedom
## Multiple R-squared:  0.8968, Adjusted R-squared:  0.8958
## F-statistic:   878 on 1 and 101 DF,  p-value: < 2.2e-16
```

And you'll see that they have the same correlation coefficient as the earlier multivariate regression

And now, let's close this exercize and plot up our multivariate regression.

```
# first we need to set a specific graphics parameter to set our plot shape.
#   "s" makes a square, "m" is the default which maximizes the plot region.

par(pty = "s") # this makes the plot square
               # (I like square plots when I plot "apples against apples")

# now a simple x-y scatterplot as before but with both axes having
#   the same range...

plot(x    = exceldata$Compressive.Strength,                        # x-values
     y    = fitted.S_v_all,                                        # y-values
     main = "Example of a Multiple Variable Regression",           # title string
     xlab = "Obs 28-dy Compressive Stren. (MPa)",                  # x-label
     ylab = "Pred 28-dy Compressive Stren. (MPa)",                 # y-label
     xlim = c(min(exceldata$Compressive.Strength,fitted.S_v_all),  # x-axis range
              max(exceldata$Compressive.Strength,fitted.S_v_all)),
     ylim = c(min(exceldata$Compressive.Strength,fitted.S_v_all),  # y-axis range
              max(exceldata$Compressive.Strength,fitted.S_v_all)),
     asp = 1                                                       # aspect ratio between
     )                                                             #    x and y scales

# plot the linear regression line

abline(reg = linear_model.S_v_Sofall,  # put the regression output here
       col = "red"                     # color it red
       )

# and we can also plot a simple one:to:one line.

abline(a   = 0,      # y-intercept
       b   = 1,      # slope
       col = "grey" # color it grey
       )
```

Example of a Multiple Variable Regression



And here we have a nice plot showing our true vs predicted values.

While here, we can do some general error metrics that may be useful..

First, the Bias... (if we are too high or too low)

```
bias = mean(fitted.S_v_all - exceldata$Compressive.Strength)

print("BIAS")
```

```
## [1] "BIAS"
```

```
print(bias)
```

```
## [1] -1.3802e-16
```

The root mean squared error (RMSE) thought he standard error of the estmate is technically the one we use here. RMSE remains a common error metric though...

```
rmse = sqrt(mean( (fitted.S_v_all - exceldata$Compressive.Strength)^2)  )

print("RMSE")
```

```
## [1] "RMSE"
```

```
print(rmse)
```

```
## [1] 2.505303
```

And finally our correlation coefficient (which is basically our coefficient of determination before the "R" is "squared")

```
r = cor(x = fitted.S_v_all,                    # the x-value
        y = exceldata$Compressive.Strength, # the y-value
        method = "pearson"                    # method of correlation
```

```r
        )

print("correlation coefficient")
```

```
## [1] "correlation coefficient"
```

```r
print(r)
```

```
## [1] 0.9470151
```

```r
print("coefficient of determination")
```

```
## [1] "coefficient of determination"
```

```r
print(r^2)
```

```
## [1] 0.8968376
```

And with that, we're done.